

Управление камерой и микрофоном

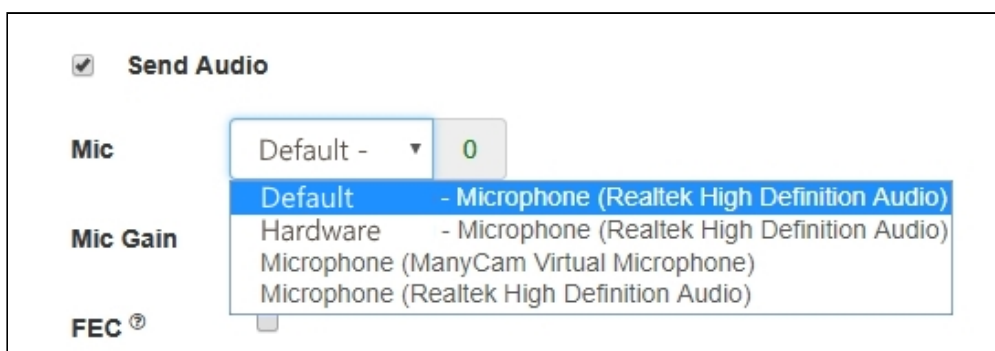
WCS позволяет настраивать камеру и микрофон в браузере. Рассмотрим, как и какими параметрами можно управлять при захвате аудио- и видеопотока, на примере веб-приложения [Media Devices](#):

[media_device_manager.html](#)

[manager.js](#)

Настройки микрофона

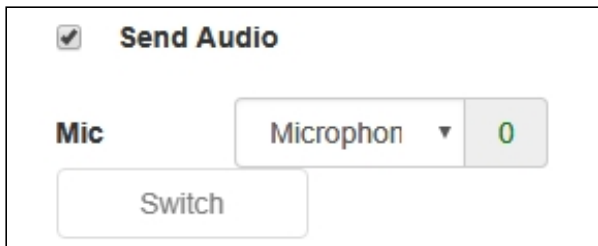
1. Выбор микрофона из списка



code:

```
Flashphoner.getMediaDevices(null, true,
MEDIA_DEVICE_KIND.INPUT).then(function (list) {
  list.audio.forEach(function (device) {
    ...
  });
  ...
}).catch(function (error) {
  $("#notifyFlash").text("Failed to get media devices");
});
```

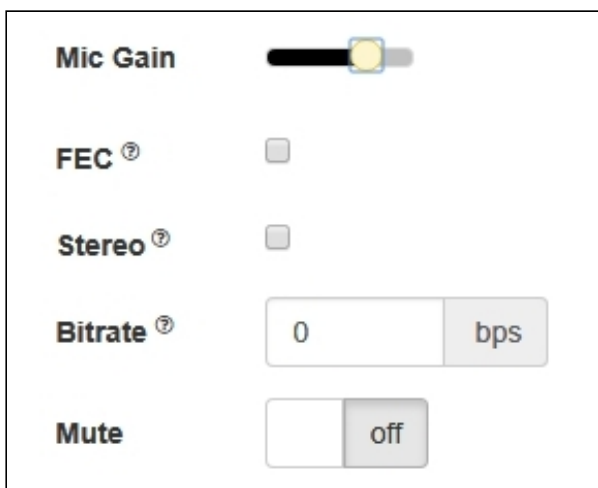
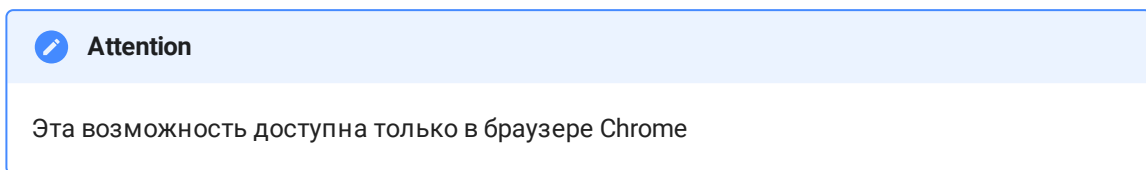
2. Переключение микрофона во время трансляции



code:

```
$("#switchMicBtn").click(function () {
  stream.switchMic().then(function (id) {
    $('#audioInput option:selected').prop('selected', false);
    $('#audioInput option[value="'+ id +'"]').prop('selected', true);
  }).catch(function (e) {
    console.log("Error " + e);
  });
}).prop('disabled', !($('#sendAudio').is(':checked')));
```

3. Регулировка усиления микрофона



code:

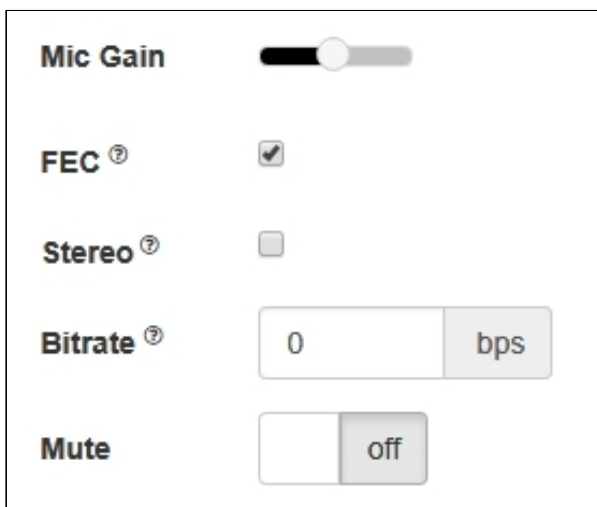
```
$("#micGainControl").slider({
  range: "min",
  min: 0,
  max: 100,
  value: currentGainValue,
  step: 10,
  animate: true,
  slide: function (event, ui) {
```

```
currentGainValue = ui.value;
if(previewStream) {
    publishStream.setMicrophoneGain(currentGainValue);
}
}
});
```

4. Включение коррекции ошибок

Attention

Эта возможность доступна только для кодека Opus



The screenshot shows a settings panel with the following controls:

- Mic Gain**: A slider control.
- FEC**: A checkbox that is checked.
- Stereo**: A checkbox that is unchecked.
- Bitrate**: A numeric input field containing '0' and a unit selector dropdown set to 'bps'.
- Mute**: A toggle switch currently set to 'off'.

code:

```
if (constraints.audio) {
    constraints.audio = {
        deviceId: $('#audioInput').val()
    };
    if ($('#fec').is(':checked'))
        constraints.audio.fec = ($('#fec').is(':checked'));
    ...
}
```

5. Установка стерео / моно режима

Mic Gain

FEC

Stereo

Bitrate bps

Mute off

code:

```
if (constraints.audio) {
  constraints.audio = {
    deviceId: $('#audioInput').val()
  };
  ...
  if ($("#sendStereoAudio").is(':checked'))
    constraints.audio.stereo = ($("#sendStereoAudio").is(':checked'));
  ...
}
```

6. Установка битрейта аудио

Attention

Значение битрейта аудио должно быть задано в бит/с, например **64000** для публикации 64 кбит/с

Mic Gain

FEC

Stereo

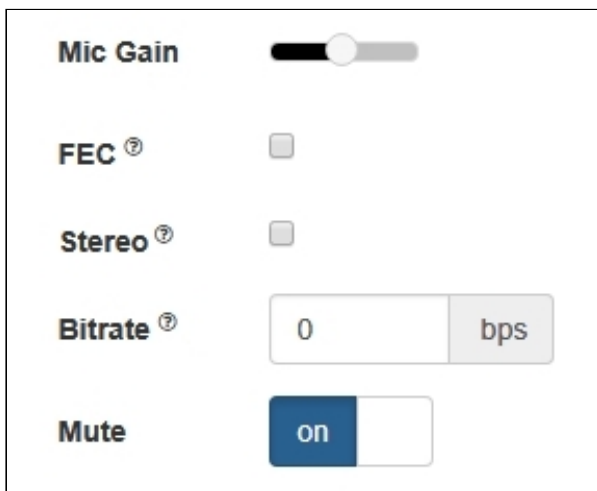
Bitrate bps

Mute off

code:

```
if (constraints.audio) {
  constraints.audio = {
    deviceId: $('#audioInput').val()
  };
  ...
  if (parseInt($('#sendAudioBitrate').val()) > 0)
    constraints.audio.bitrate = parseInt($('#sendAudioBitrate').val());
}
```

7. Отключение микрофона



The image shows a settings panel for audio configuration. It includes a slider for 'Mic Gain', checkboxes for 'FEC' and 'Stereo', a text input for 'Bitrate' with a unit selector set to 'bps', and a toggle switch for 'Mute' which is currently turned on.

code:

```
if ($("#muteAudioToggle").is(":checked")) {
  muteAudio();
}
```

Настройки камеры

1. Выбор камеры



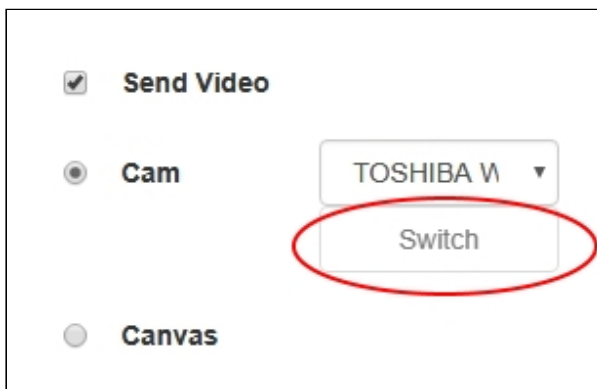
code:

```
Flashphoner.getMediaDevices(null, true,
MEDIA_DEVICE_KIND.INPUT).then(function (list) {
  ...
  list.video.forEach(function (device) {
    ...
  });
}).catch(function (error) {
  $("#notifyFlash").text("Failed to get media devices");
});
```

Если необходимо выбрать только камеру, не запрашивая доступ к аудиоустройствам, необходимо вызвать функцию `getMediaDevices()` с явным указанием ограничений

```
Flashphoner.getMediaDevices(null, true, MEDIA_DEVICE_KIND.INPUT, {video:
true, audio: false}).then(function (list) {
  ...
  list.video.forEach(function (device) {
    ...
  });
}).catch(function (error) {
  $("#notifyFlash").text("Failed to get media devices");
});
```

2. Переключение камер



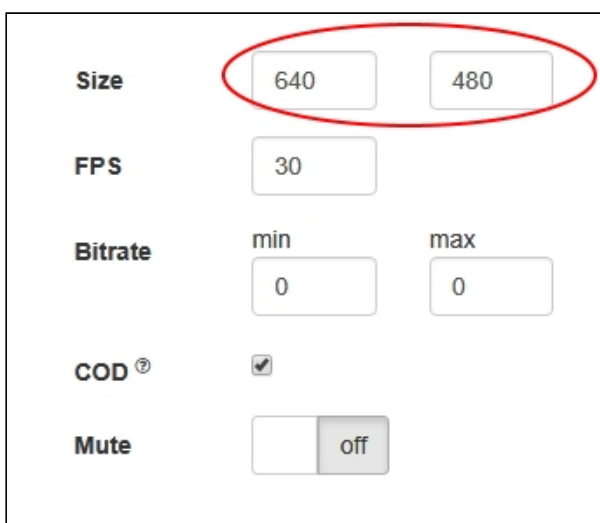
code:

```
$("#switchBtn").text("Switch").off('click').click(function () {
    stream.switchCam().then(function(id) {
        $('#videoInput option:selected').prop('selected', false);
        $('#videoInput option[value="'+ id +'"').prop('selected', true);
    }).catch(function(e) {
        console.log("Error " + e);
    });
});
```

Переключение камеры может осуществляться "на лету", во время трансляции потока. Переключение работает в следующем порядке:

- На ПК камеры переключаются в том порядке, в каком они определены в менеджере устройств операционной системы
- На Android при использовании браузера Chrome по умолчанию выбирается фронтальная камера, при использовании браузера Firefox - тыловая камера
- На iOS в браузере Safari по умолчанию выбирается фронтальная камера, но в выпадающем списке при выборе камеры первой указана тыловая камера

3. Установка разрешения видео



The image shows a settings interface for video recording. The 'Size' section has two input fields: '640' and '480', which are circled in red. Below this, the 'FPS' is set to '30'. The 'Bitrate' section has 'min' and 'max' fields, both set to '0'. The 'COD' section has a checked checkbox. The 'Mute' section has a toggle switch set to 'off'.

code:

```
constraints.video = {
    deviceId: $('#videoInput').val(),
    width: parseInt($('#sendWidth').val()),
    height: parseInt($('#sendHeight').val())
};
if (Browser.isSafariWebRTC() && Browser.isiOS() &&
Flashponer.getMediaProviders()[0] === "WebRTC") {
    constraints.video.deviceId = {exact: $('#videoInput').val()};
}
```

4. Установка FPS

| | | |
|-------------------------|-------------------------------------|--------------------------------------|
| Size | <input type="text" value="320"/> | <input type="text" value="240"/> |
| FPS | <input type="text" value="15"/> | |
| Bitrate | min <input type="text" value="0"/> | max <input type="text" value="0"/> |
| COD [®] | <input checked="" type="checkbox"/> | |
| Mute | <input type="checkbox"/> | <input type="checkbox" value="off"/> |

code:

```
if (constraints.video) {  
  ...  
  if (parseInt($('#fps').val()) > 0)  
    constraints.video.frameRate = parseInt($('#fps').val());  
}
```

5. Установка битрейта видео

Attention

Значение битрейта видео должно задаваться в кбит/с

| | | |
|-------------------------|--------------------------------------|---------------------------------------|
| Size | <input type="text" value="320"/> | <input type="text" value="240"/> |
| FPS | <input type="text" value="30"/> | |
| Bitrate | min <input type="text" value="500"/> | max <input type="text" value="1000"/> |
| COD [®] | <input checked="" type="checkbox"/> | |
| Mute | <input type="checkbox"/> | <input type="checkbox" value="off"/> |

code:

```
if (constraints.video) {
  ...
  if (parseInt($('#sendVideoMinBitrate').val()) > 0)
    constraints.video.minBitrate =
parseInt($('#sendVideoMinBitrate').val());
  if (parseInt($('#sendVideoMaxBitrate').val()) > 0)
    constraints.video.maxBitrate =
parseInt($('#sendVideoMaxBitrate').val());
  ...
}
```

6. Установка CPU Overuse Detection

| | | |
|------------------------|----------------------------------|--------------------------------------|
| Size | <input type="text" value="320"/> | <input type="text" value="240"/> |
| FPS | <input type="text" value="30"/> | |
| Bitrate | min | max |
| | <input type="text" value="0"/> | <input type="text" value="0"/> |
| COD[®] | <input type="checkbox"/> | |
| Mute | <input type="checkbox"/> | <input type="checkbox" value="off"/> |

code:

```
if (!$("#cpuOveruseDetection").is(':checked')) {
  mediaConnectionConstraints = {
    "mandatory": {
      googCpuOveruseDetection: false
    }
  }
}
```

7. Отключение камеры

| | | |
|-------------------------|-------------------------------------|----------------------------------|
| Size | <input type="text" value="320"/> | <input type="text" value="240"/> |
| FPS | <input type="text" value="30"/> | |
| Bitrate | min | max |
| | <input type="text" value="0"/> | <input type="text" value="0"/> |
| COD [?] | <input checked="" type="checkbox"/> | |
| Mute | <input checked="" type="checkbox"/> | |

code:


```
if ($("#muteVideoToggle").is(":checked")) {
    muteVideo();
}
```

Тестирование захвата с камеры и микрофона локально

Локальное тестирование захвата с микрофона и камеры предназначено для того, чтобы проверить работоспособность микрофона и камеры в браузере, не отправляя поток на сервер.


Media Devices

Local



➔

Preview



WCS

Play audio

code:

```

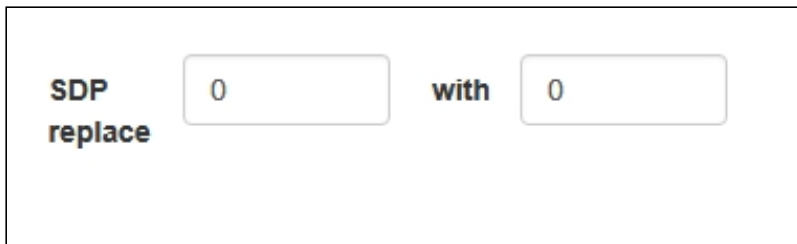
function startTest() {
    Flashphoner.getMediaAccess(getConstraints(), localVideo).then(function
    (disp) {
        $("#testBtn").text("Release").off('click').click(function () {
            $(this).prop('disabled', true);
            stopTest();
        }).prop('disabled', false);

        window.AudioContext = window.AudioContext ||
        window.webkitAudioContext;
        if (Flashphoner.getMediaProviders()[0] == "WebRTC" &&
        window.AudioContext) {
            for (i = 0; i < localVideo.children.length; i++) {
                if (localVideo.children[i] &&
                localVideo.children[i].id.indexOf("-LOCAL_CACHED_VIDEO") != -1) {
                    var stream = localVideo.children[i].srcObject;
                    audioContextForTest = new AudioContext();
                    var microphone =
                    audioContextForTest.createMediaStreamSource(stream);
                    var javascriptNode =
                    audioContextForTest.createScriptProcessor(1024, 1, 1);
                    microphone.connect(javascriptNode);
                    javascriptNode.connect(audioContextForTest.destination);
                    javascriptNode.onaudioprocess = function (event) {
                        var inpt_L = event.inputBuffer.getChannelData(0);
                        var sum_L = 0.0;
                        for (var i = 0; i < inpt_L.length; ++i) {
                            sum_L += inpt_L[i] * inpt_L[i];
                        }
                        $("#micLevel").text(Math.floor(Math.sqrt(sum_L /
                        inpt_L.length) * 100));
                    }
                }
            }
        } else if (Flashphoner.getMediaProviders()[0] == "Flash") {
            micLevelInterval = setInterval(function () {
                $("#micLevel").text(disp.children[0].getMicrophoneLevel());
            }, 500);
            testStarted = true;
        }).catch(function (error) {
            $("#testBtn").prop('disabled', false);
            testStarted = false;
        });
    });
}

```

Замена отдельных параметров SDP

При публикации потока предусмотрена возможность замены параметров SDP. В примере в поле **SDP replace** указывается шаблон поиска параметра, который нужно заменить, в поле **with** указывается новое значение параметра.



Для замены параметров SDP используется callback-функция, которая должна быть указана при создании потока в параметре `sdpHook` метода `createStream()`:

создание потока [code](#)

```
publishStream = session.createStream({
  name: streamName,
  display: localVideo,
  cacheLocalResources: true,
  constraints: constraints,
  mediaConnectionConstraints: mediaConnectionConstraints,
  sdpHook: rewriteSdp,
  ...
})
```

функция `rewriteSdp()` [code](#)

```
function rewriteSdp(sdp) {
  var sdpStringFind = $("#sdpStringFind").val().replace('\r\n', '\r\n');
  var sdpStringReplace =
    $("#sdpStringReplace").val().replace('\r\n', '\r\n');
  if (sdpStringFind != 0 && sdpStringReplace != 0) {
    var newSDP = sdp.sdpString.toString();
    newSDP = newSDP.replace(new RegExp(sdpStringFind, "g"),
      sdpStringReplace);
    return newSDP;
  }
  return sdp.sdpString;
}
```

Увеличение битрейта публикуемого видео в браузере Chrome

Замена параметров SDP позволяет увеличить битрейт публикуемого видео. Для этого необходимо при публикации H264 заменить параметр `a=fmt` по шаблону

```
a=fmt:(.*) (.*)
```

на

```
a=fmt:$1 $2;x-google-min-bitrate=2500
```

Здесь `2500` - битрейт в килобитах в секунду.

Подобным образом можно указать битрейт видео на старте (атрибут `x-google-start-bitrate`) и ограничить максимальный битрейт (атрибут `x-google-max-bitrate`). Отметим, что, если указать только минимальный битрейт, то выше 2500 кбит/с битрейт поднять не удастся, возможно, по умолчанию в Chrome зафиксирован максимальный битрейт на уровне 2500 кбит/с. Если необходимо использовать более высокие значения, например, для трансляции потока высокого разрешения, должны быть указаны и минимальное, и максимальное значения:

```
a=fmtp:$1 $2;x-google-max-bitrate=7000;x-google-min-bitrate=3000
```

В этом случае браузер будет держать битрейт при публикации потока в пределах от 3000 до 7000 кбит/с.

При публикации потока VP8 необходимо заменить

```
a=rtpmap:(.*) VP8/90000\r\n
```

на

```
a=rtpmap:$1 VP8/90000\r\na=fmtp:$1 x-google-min-bitrate=3000;x-google-max-bitrate=7000\r\n
```

Возможность управления битрейтом доступна только в браузере Chrome.

Задание пропускной способности канала

Замена параметров SDP позволяет задать пропускную способность канала при публикации потока. Для этого необходимо при публикации заменить параметр `c=IN` по шаблону

```
c=IN (.)\r\n
```

на

```
c=IN $1\r\nb=AS:10000\r\n
```

Установка используемых кодеков

При публикации потока предусмотрена возможность убрать из WebRTC SDP кодеки, которые не должны использоваться при публикации данного потока, например:

```
publishStream = session.createStream({  
  ...  
  stripCodecs: ["h264,H264,flv,mpv"]  
}).on(STREAM_STATUS.PUBLISHING, function (publishStream) {
```

```
...
});
publishStream.publish();
```

Данная возможность полезна, в частности, для обхода проблем браузера с каким-либо кодеком. Например, если в браузере не работает H264, можно отключить его и перейти на VP8 при публикации WebRTC.

Управление выводом звука

При воспроизведении потока можно выбрать (и переключить "на лету") устройство вывода звука в браузерах на базе Chrome.



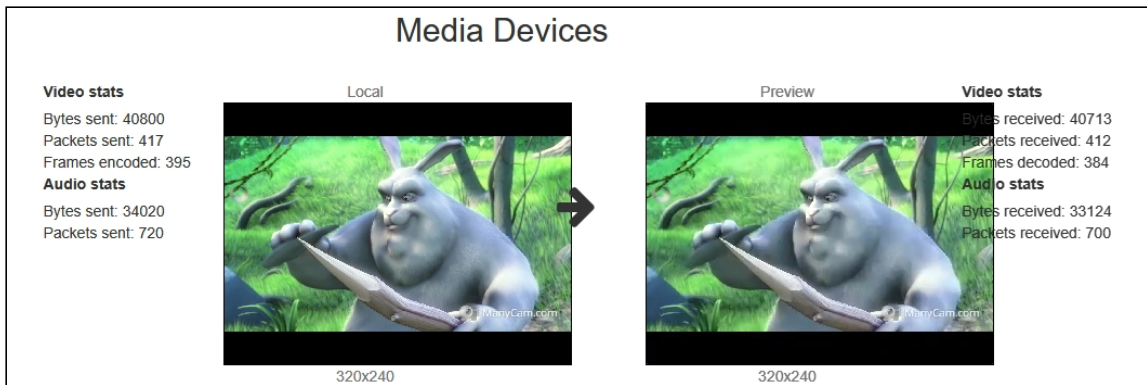
code:

```
Flashphoner.getMediaDevices(null, true,
MEDIA_DEVICE_KIND.OUTPUT).then(function (list) {
  list.audio.forEach(function (device) {
    ...
  });
}).catch(function (error) {
  $('#audioOutputForm').remove();
});
```

Отметим, что в браузерах Firefox и Safari нельзя получить список устройств вывода, поэтому данная функция в них не работает

Отображение WebRTC-статистики

При публикации и воспроизведении потока клиентское приложение может получить WebRTC-статистику в соответствии со [стандартом](#). Эта статистика может быть отображена в браузере, например:



1. Отображение статистики при публикации потока

`Stream.getStats()` [code](#)

```
publishStream.getStats(function (stats) {
  if (stats && stats.outboundStream) {
    if (stats.outboundStream.video) {
      showStat(stats.outboundStream.video, "outVideoStat");
      let vBitrate = (stats.outboundStream.video.bytesSent -
videoBytesSent) * 8;
      if ($('#outVideoStatBitrate').length == 0) {
        let html = "<div>Bitrate: " + "<span
id='outVideoStatBitrate' style='font-weight: normal'>" + vBitrate + "
</span>" + "</div>";
        $('#outVideoStat').append(html);
      } else {
        $('#outVideoStatBitrate').text(vBitrate);
      }
      videoBytesSent = stats.outboundStream.video.bytesSent;
      ...
    }

    if (stats.outboundStream.audio) {
      showStat(stats.outboundStream.audio, "outAudioStat");
      let aBitrate = (stats.outboundStream.audio.bytesSent -
audioBytesSent) * 8;
      if ($('#outAudioStatBitrate').length == 0) {
        let html = "<div>Bitrate: " + "<span
id='outAudioStatBitrate' style='font-weight: normal'>" + aBitrate + "
</span>" + "</div>";
        $('#outAudioStat').append(html);
      } else {
        $('#outAudioStatBitrate').text(aBitrate);
      }
      audioBytesSent = stats.outboundStream.audio.bytesSent;
    }
  }
  ...
});
```

2. Отображение статистики при воспроизведении потока

`Stream.getStats()` [code](#)

```

previewStream.getStats(function (stats) {
  if (stats && stats.inboundStream) {
    if (stats.inboundStream.video) {
      showStat(stats.inboundStream.video, "inVideoStat");
      let vBitrate = (stats.inboundStream.video.bytesReceived -
videoBytesReceived) * 8;
      if ($('#inVideoStatBitrate').length == 0) {
        let html = "<div>Bitrate: " + "<span
id='inVideoStatBitrate' style='font-weight: normal'>" + vBitrate + "
</span>" + "</div>";
        $('#inVideoStat').append(html);
      } else {
        $('#inVideoStatBitrate').text(vBitrate);
      }
      videoBytesReceived = stats.inboundStream.video.bytesReceived;
      ...
    }

    if (stats.inboundStream.audio) {
      showStat(stats.inboundStream.audio, "inAudioStat");
      let aBitrate = (stats.inboundStream.audio.bytesReceived -
audioBytesReceived) * 8;
      if ($('#inAudioStatBitrate').length == 0) {
        let html = "<div style='font-weight: bold'>Bitrate: " + "
<span id='inAudioStatBitrate' style='font-weight: normal'>" + aBitrate + "
</span>" + "</div>";
        $('#inAudioStat').append(html);
      } else {
        $('#inAudioStatBitrate').text(aBitrate);
      }
      audioBytesReceived = stats.inboundStream.audio.bytesReceived;
    }
    ...
  }
});

```

Управление параметрами картинки при публикации потока

При публикации видео потока с помощью граничных условий (constraints) можно управлять разрешением картинки и частотой кадров

Управление разрешением картинки

Разрешение картинки можно указать точно

```
constraints = {audio:true, video:{width:320,height:240}}
```

Однако, в некоторых случаях требуется указать диапазон для ширины и высоты


```
constraints = {audio:true, video:{width:{min:160,max:320},height:{min:120,max:240}}}
```

Для некоторых браузеров, например, iOS Safari, необходимо указать точные значения в виде диапазона (в последних [версиях](#) это обрабатывается на уровне WebSDK)

```
constraints = {audio:true, video:{width:{min:320,max:320},height:{min:240,max:240}}}
```

Управление частотой кадров

Частота кадров может быть указана точно

```
constraints = {audio:true, video:{frameRate:30}}
```

или в виде диапазона

```
constraints = {audio:true, video:{frameRate:{min:15,max:30}}}
```

В некоторых случаях, например, если веб-камера поддерживает 24 fps, при точном указании 30 fps публикация может завершиться ошибкой. В таком случае нужно задать частоту кадров как идеальную

```
constraints = {audio:true, video:{frameRate:{ideal:30}}}
```

Переключение между потоками с веб-камеры и с экрана во время трансляции

При организации вебинаров возникает необходимость переключаться между потоками, захваченными с веб-камеры ведущего и с экрана, во время трансляции. В идеале, переключение должно быть бесшовным и с сохранением звуковой дорожки с микрофона ведущего. В последних версиях WebSDK реализована такая возможность для браузеров Chrome и Firefox, рассмотрим пример использования в приложении Media Devices.

Media Devices


Video stats

Bytes sent: 389875
Packets sent: 430
Frames encoded: 183

Audio stats


Bytes sent: 33404
Packets sent: 420

Local



640x480

Preview



640x480

WCS PUBLISHING

Screen share off

Size

FPS

Bitrate min max

COD

Mute off

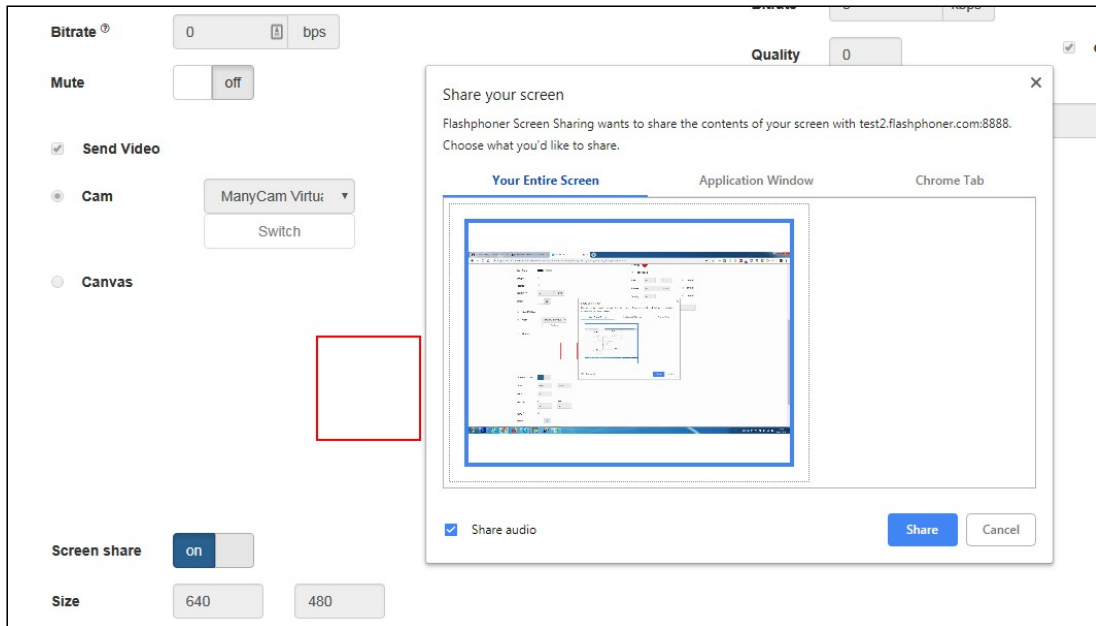
- Во время трансляции потока с выбранных камеры и микрофона, при установке переключателя **Screen share** в положение **on** будет вызвана функция `switchToScreen()`
`Stream.switchToScreen()` code

```
function switchToScreen() {
  if (publishStream) {
    $('#switchBtn').prop('disabled', true);
    $('#videoInput').prop('disabled', true);

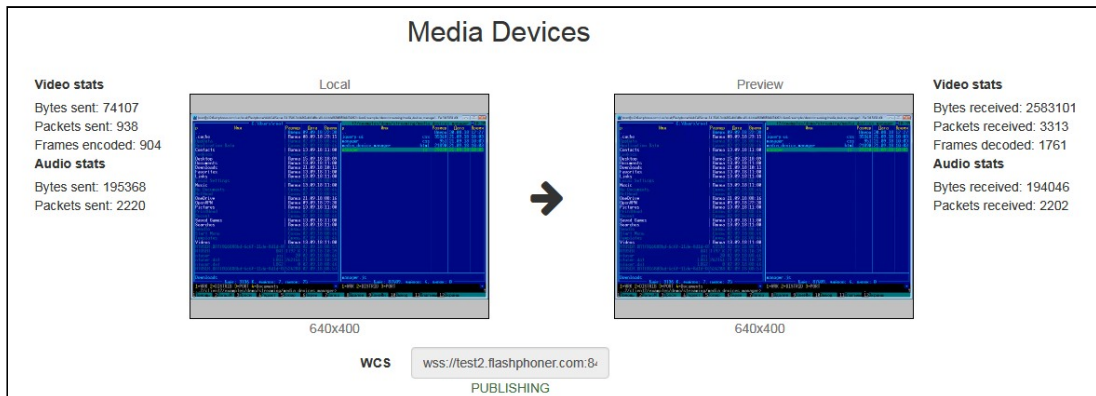
    publishStream.switchToScreen($('#mediaSource').val()).catch(function () {
      $('#screenShareToggle').removeAttr("checked");
      $('#switchBtn').prop('disabled', false);
      $('#videoInput').prop('disabled', false);
    });
  }
}
```

В данную функцию передается источник потока (экран).

2. Затем пользователь должен выбрать весь экран или окно программы для трансляции:



3. На сервер транслируется поток с экрана



При этом источник трансляции звука не меняется.

4. Для возврата к трансляции потока с веб-камеры вызывается функция

`switchToCam()` `Stream.switchToCam()` code

```
function switchToCam() {
  if (publishStream) {
    publishStream.switchToCam();
    $('#switchBtn').prop('disabled', false);
    $('#videoInput').prop('disabled', false);
  }
}
```

Ограничения

1. Переключение трансляции работает только в браузерах Chrome и Firefox
2. Невозможно переключить веб-камеру в то время, когда транслируется экран

3. Переключение работает только в том случае, если первым опубликован поток с веб-камеры.

Известные проблемы

1. Не работает переключение микрофона в браузере Safari

Симптомы

Не переключается микрофон при помощи метода WebSDK `switchMic()`.

Решение

Использовать другой браузер, поскольку Safari всегда использует микрофон `sound input`, выбранный в настройках звука системы (для входа необходимо зажать клавишу `Option` и щелкнуть по иконке звука в меню). После выбора другого микрофона требуется перезагрузка Mac.

Если не работает микрофон Logitech USB camera (когда выбран в `sound input`), может помочь изменение `format / sample rate` в `Audio MIDI Setup` и перезагрузка.

2. iOS Safari зависает на воспроизведении, если публикующий переключает камеру

Симптомы

При переключении камеры воспроизведение публикуемого потока в браузере iOS Safari зависает.

Решение

Включить транскодинг при помощи параметра в файле `flashphoner.properties`

```
disable_streaming_proxy=true
```

или указав фиксированное разрешение для плеера при воспроизведении

```
session.createStream({constraints:{audio:true,video:{width:320,height:240}}}).play();
```

